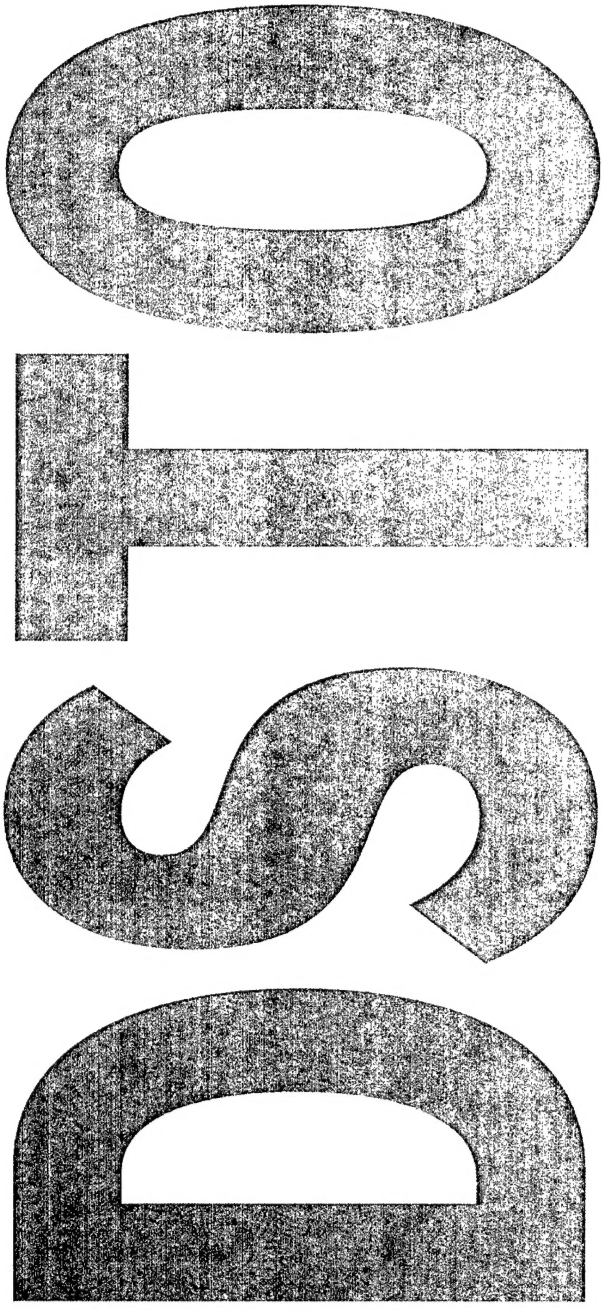
Australian Government

Department of Defence

Defence Science and
Technology Organisation



Visualising Line-of-Sight Information in Matlab

David Mewett, Damian Hall and
Adam Davies

DSTO-GD-0397

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

BEST AVAILABLE COPY

20040617 037

Australian Government
Department of Defence
Defence Science and
Technology Organisation

Visualising Line-of-Sight Information in Matlab

David Mewett, Damian Hall and Adam Davies

Intelligence, Surveillance and Reconnaissance Division
Information Sciences Laboratory

DSTO-GD-0397

ABSTRACT

There are many ways to visualise areas that are within line-of-sight of a sensor or communications transmitter. This document describes how to use visualisation and mapping commands in Matlab to produce a variety of line-of-sight maps to suit customers'' requirements.

RELEASE LIMITATION

Approved for public release

AQ F04-09-0997

Published by

DSTO Information Sciences Laboratory
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

© Commonwealth of Australia 2004
AR-013-059
March 2004

APPROVED FOR PUBLIC RELEASE

Visualising Line-of-Sight Information in Matlab

Executive Summary

This document explains how to use combinations of commands from Matlab and the Matlab Mapping Toolbox to display line-of-sight information in maps. Line-of-sight information may be required for analysis in several areas relevant to Defence, including surveillance, reconnaissance and communications. The techniques described here were developed for applications including radar siting and visibility from Unmanned Aerial Vehicles. These techniques are a mixture of methods described as examples in Matlab documentation and methods developed in response to particular problems.

The intention of this document is to give examples of a range of display options that are possible in Matlab, and to give clear instructions of how to produce them.

Contents

1. INTRODUCTION 1

2. MAPPING TOOLBOX CONVENTIONS 1
   - 2.1 Representation of Maps 1
   - 2.2 Line-of-Sight Functions and Radio Waves 1

3. DISPLAYING MAPS 2
   - 3.1 Maps Without Terrain Information 2
   - 3.2 Determining Colour Settings 2
   - 3.3 Maps With Terrain Displayed by Colour 3
   - 3.4 Maps with terrain displayed by contour lines 5

4. DISPLAYING LINE-OF-SIGHT LIMITS 6
   - 4.1 Line-of-Sight Data 6
   - 4.2 Line-of-Sight Maps 6
     - 4.2.1 Simple display 6
     - 4.2.2 Draping line-of-sight map over terrain 6
   - 4.3 Line-of-Sight Overlaid on Other Maps 8
     - 4.3.1 Transparent surface indexing colour map 8
     - 4.3.2 Transparent surface using arbitrary colours 9
     - 4.3.3 Arbitrary colour projected onto terrain map 11
     - 4.3.4 Outline of line-of-sight 12
     - 4.3.5 Floating transparent surfaces using arbitrary colours 13
   - 4.4 Distance Markers 15
   - 4.5 Three-dimensional Displays 15
     - 4.5.1 Adjustments for three-dimensional views 15
     - 4.5.2 OpenGL Issues 16

5. CONCLUSION 16

APPENDIX A: REMOVING ARTEFACTS 17
   - A.1. Artefacts in Line-of-Sight Maps 17
   - A.2. Removing Artefacts 17

1. Introduction

Line-of-sight information is important for defence applications such as surveillance and communications. The Matlab Mapping Toolbox (The Math Works Inc., Natick, MA, USA) includes routines to calculate line-of-sight based on terrain elevation data. The results can be displayed on two-dimensional and three-dimensional maps in various formats.

Examples of Matlab commands in this document are printed in `Courier` typeface. All commands are from Release 13 (Matlab 6.5, Mapping Toolbox 1.3). Most are common to earlier releases, although some may not be.

2. Mapping Toolbox Conventions

2.1 Representation of Maps

Map information may be characterised as vector or matrix (raster) information. Vector maps consist of latitude-longitude pairs that define features such as coastlines, with individual line segments separated by 'not-a-number' (`NaN`) values. Matrix maps describe information relating to an area. Two different types of data structure are used to represent matrix maps in the Mapping Toolbox. The structure that is used for line-of-sight calculations, called a *regular matrix map*, consists of:

1. the *map*, a matrix of values (such as terrain elevation data) on a regular grid;
2. the *map legend*, a vector containing the resolution of the map and the latitude and longitude of the north-west corner of the map.

Refer to the *Mapping Toolbox Users' Guide* for details on the regular matrix map structure. Version 2.0 of the Toolbox (and later) uses different terminology; in particular, *data grid* instead of *map*, and *referencing vector* instead of *map legend*.

2.2 Line-of-Sight Functions and Radio Waves

Atmospheric refraction of radio waves beyond the optical horizon can be accounted for by treating the problem as straight line propagation on a sphere with 4/3 of the Earth's radius. This is set via the `EffectiveRadius` input to the two line-of-sight functions, `los2` (line-of-sight between two points) and `viewshed` (all points visible to an observer). The Earth's radius is accessed via the `almanac` command.

The diffraction of radio waves around obstructions is not modelled in the line-of-sight functions.

The line-of-sight map returned by `viewshed` does not take the viewing angle of any sensors into account, either. The algorithm simply determines whether straight lines can be drawn between pairs of points without passing through the terrain.

3. Displaying Maps

3.1 Maps Without Terrain Information

Coastal outline maps can be displayed using the `worldmap` function. Landmasses can be shown as filled outlines (`'patch'` or `'patchonly'` options in `worldmap`) or outlines only (`'line'` or `'lineonly'`). If patches are used, `worldmap` uses the same colour for all islands in a given country, but chooses the colour arbitrarily. The `colormap` command can be used to set the colour of a displayed map, e.g. to show dark green land with a coast outline (Figure 1):

```
worldmap([-13,-11], [130,133], 'patch')
colormap([0,0.4,0.2])
```

The default colour for the sea is white, but this colour can be changed by setting the map axes' frame face colour property, e.g. to show the sea using cyan (Figure 1):

```
setm(gca, 'FFaceColor', 'cyan')
```

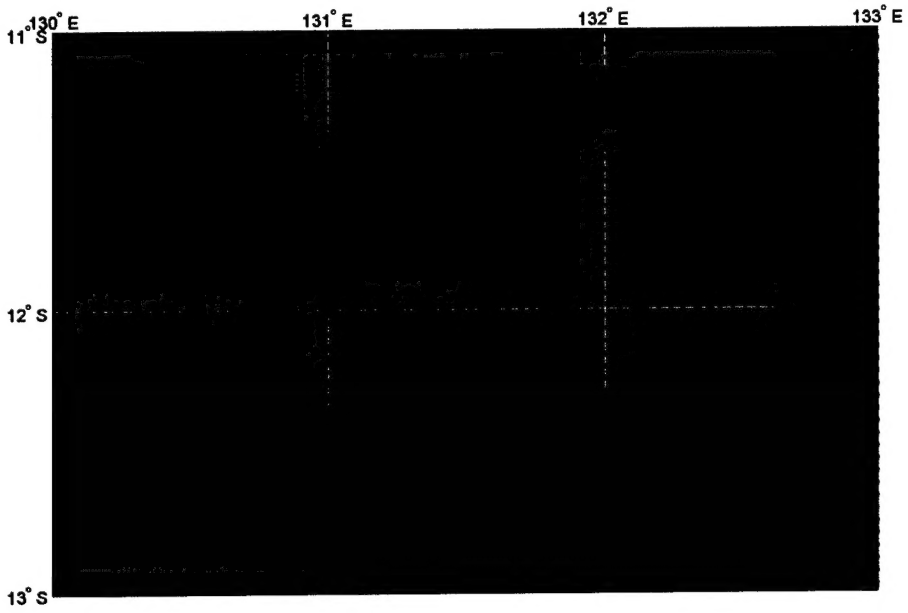


Figure 1 Basic map without terrain information.

3.2 Determining Colour Settings

It can be difficult to know how to set arbitrary (un-named) colours by the three-element 'RGB' vector used by the `colormap` and `setm` commands. The colours of patches can be set interactively from the figure window, but each patch must be

selected separately. One method of determining the RGB code for a desired colour is to generate a simple line plot, change the line to a custom colour, then interrogate the `color` property of the line:

```
h = plot(1:5, 'LineWidth', 7);
⟨ edit plot to select colour ⟩
get(h, 'Color')
```

3.3 Maps With Terrain Displayed by Colour

Terrain elevation data in the Matlab workspace is also easily displayed with the `worldmap` command. Suppose we have a map of elevation data loaded as `map` and `maplegend`. If a flat-coloured sea is desired, all values below an elevation of zero should be set to a constant negative value. The terrain map is displayed with a flat sea and coastline using:

```
map(map <= 0) = -1;
worldmap(map,maplegend), demcmap(map)
cbh = colorbar('horiz');
set(get(cbh,'XLabel'), 'String', 'Elevation (m)')
```

which also places a labelled colour bar under the map as a key to elevation data (Figure 2). It is necessary to use the graphics object handle `cbh` to set the label because the colour bar's `xlabel` property can not be accessed via the figure window.

The problem with this simple method of displaying terrain data is that there may be large discrepancies between coastlines, depending on the location (Figure 2). This can be corrected by placing the elevation map over a filled coastal outline (Figure 3):

```
landmap = map; landmap(landmap <= 0) = NaN;
worldmap(landmap,maplegend,'patch'); demcmap(map)
setm(gca,'FFaceColor','cyan')
meshm(landmap,maplegend,size(landmap),landmap)
gridm reset, setm(gca,'gcolor','k')
```

In this case, the colour of the background map is set by the `demcmap` command; the 'DEM' colour map is then automatically applied to the terrain data plotted by `meshm`. The final line resets the grid to its default position above the map and changes its colour to black.

The key operation in combining the flat map and terrain map is to replace the sea in the terrain map with 'not-a-number' (`NaN`) values. The terrain data values index into the 'DEM' colour map, so a `NaN` specifies 'no colour'. This allows the background map to show through the terrain map in these locations.

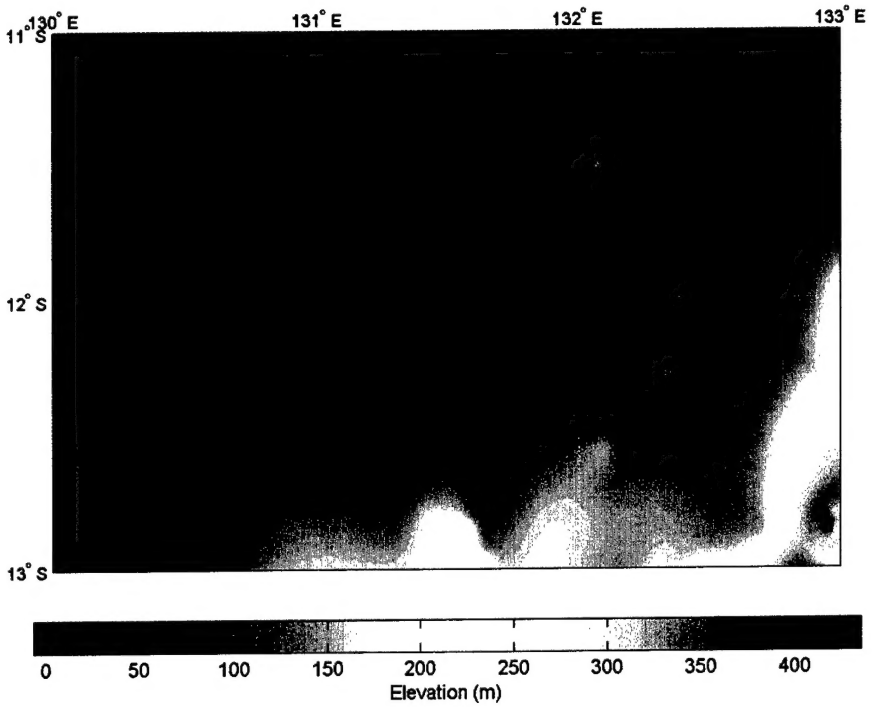


Figure 2 Simple display of terrain data (DTED level 0) with coastline.

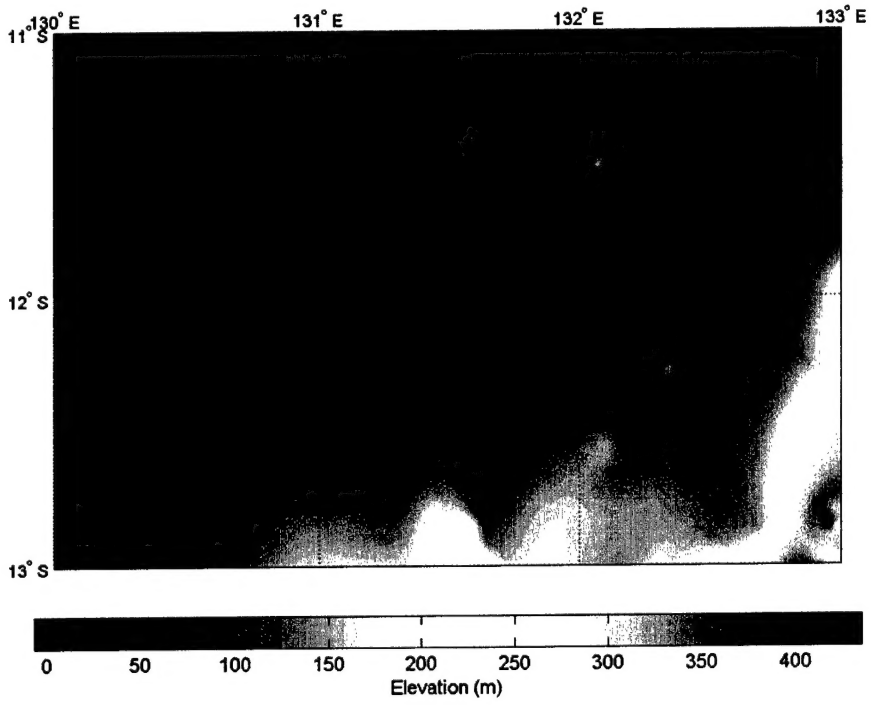


Figure 3 Improved display of terrain data with coastline.

3.4 Maps with terrain displayed by contour lines

Terrain can also be displayed by contour maps. For example, we might choose to display the map of Figure 3 using black contour lines separated by 20 m (Figure 4):

```
contourm(map, maplegend, 20:20:max(map(:)), 'k')
```

There are several options for displaying contour lines, including applying an elevation colour map. However, this might be visually confusing when line-of-sight information is overlaid on the map.

4. Displaying Line-of-Sight Limits

4.1 Line-of-Sight Data

The data used in the following examples are from line-of-sight `viewshed` results (in matrices `vmapA` and `vmapB` with corresponding legend `vmaplegend`) for a low-altitude flying target observed from two different radar sites.

4.2 Line-of-Sight Maps

4.2.1 Simple display

The simplest method of showing line-of-sight information is to display the regular matrix map `vismap` returned by the `viewshed` function. This is particularly appropriate if determining the region of terrain with line-of-sight to a flying target.

So that the resulting map only contains colours corresponding to 'visible' and 'obscured', the results for the two sites were combined using a logical OR operation:

```
vmaps = double(vmapA | vmapB);
```

Terrain information can be represented via contour lines over a line-of-sight map displayed in a plane (Figure 4):

```
worldmap(vmaps, vmaplegend)
colormap(cool(2))
h_cb = lcolorbar({'obscured','visible'});
set(h_cb, 'Position', [0.875, 0.45, 0.02, 0.1])    % resize
contourm(map, maplegend, 20:20:max(map(:)), 'k')
```

4.2.2 Draping line-of-sight map over terrain

The line-of-sight map can also be draped over the terrain, with lighting used to highlight geographical features (Figure 5):

```
worldmap(map, maplegend, 'line')
meshm(vmaps, vmaplegend, size(map), map)
colormap(cool(2))
h_light = lightangle(240,40);
material dull, lighting phong
```

The `lightmui` tool can be used to interactively change the position of the light source created by `lightangle` and so achieve different lighting effects. The `lightm` command can be used to specify the light source position by latitude and longitude, but this may give errors if the desired position is not in the current map axis.

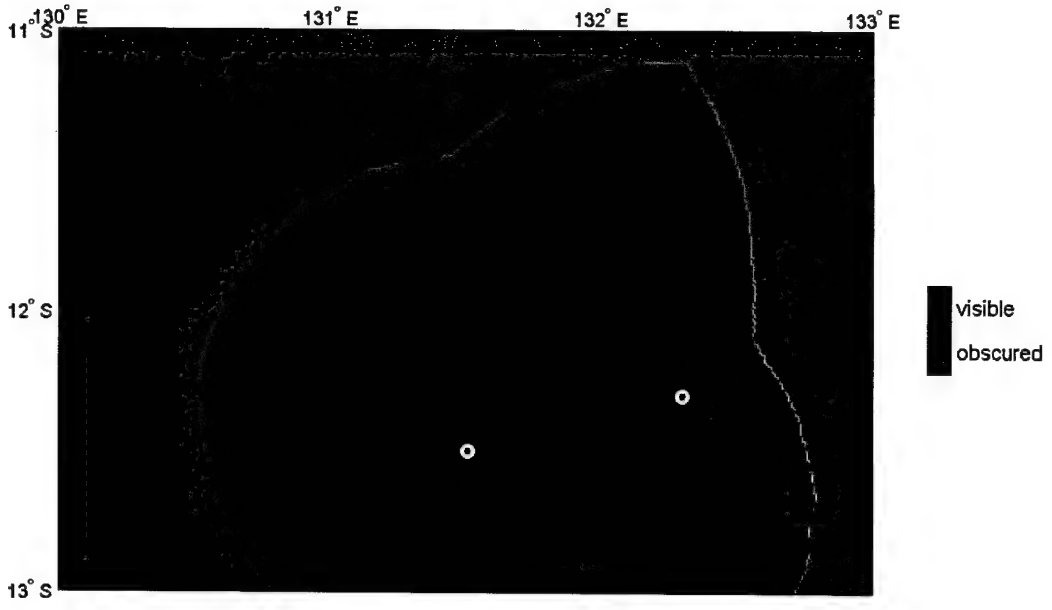


Figure 4 Line-of-sight map with terrain displayed using contour lines spaced by 20 m.

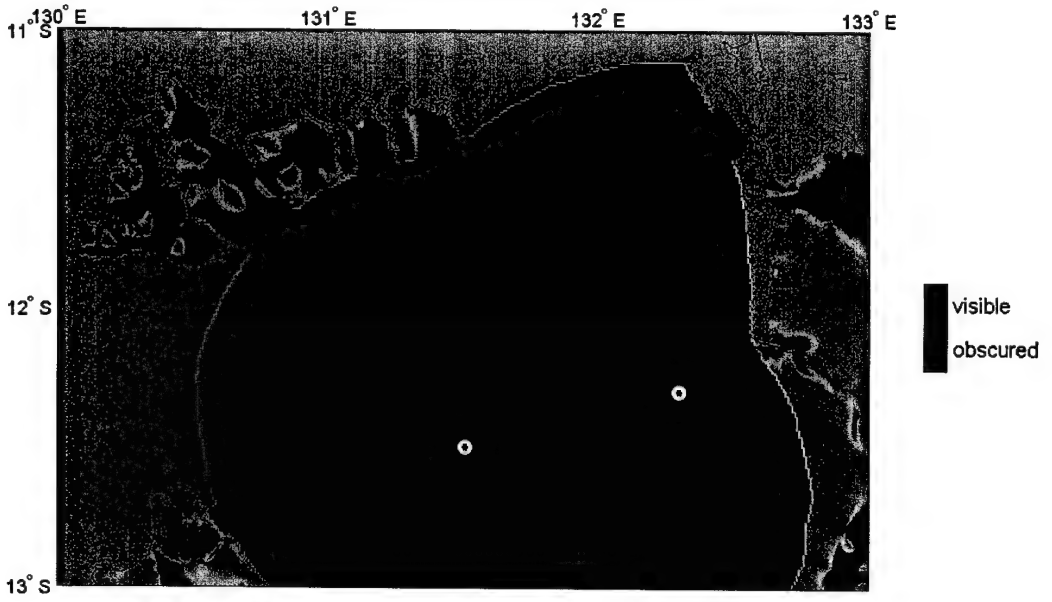


Figure 5 Line-of-sight map with terrain displayed using lighting effects.

In Figure 5, the observer positions were plotted using `plot3m` instead of `plotm`, with the altitudes at the sites determined from the terrain map using the `ltln2val` function.

4.3 Line-of-Sight Overlaid on Other Maps

4.3.1 Transparent surface indexing colour map

The line-of-sight map can be displayed as a transparent overlay above an existing terrain map by using OpenGL graphics rendering. All obscured positions are set to `NaN` (so that they do not index into the colour map) and the line-of-sight map is plotted as a surface at some altitude above the terrain map. This altitude is set via the `alt` matrix input to the `meshm` function. The `alpha` property of this surface is set between 0 and 1 to make it transparent.

```
vmaps(vmaps <= 0) = NaN;
alt = 500 * ones(size(vmaps));
h_mesh = meshm(vmaps, vmaplegend, size(map), alt);
set(gcf, 'renderer', 'OpenGL')
alpha(h_mesh, 0.5);
```

The surface will appear dark green because the ones in the line-of-sight map `vmaps` index the lowest value in the 'DEM' colour map (Figure 6). If `vmaps` were scaled to different values, then different colours would be used e.g. a value of 400 would be equivalent to brown for the colour map in Figure 6.

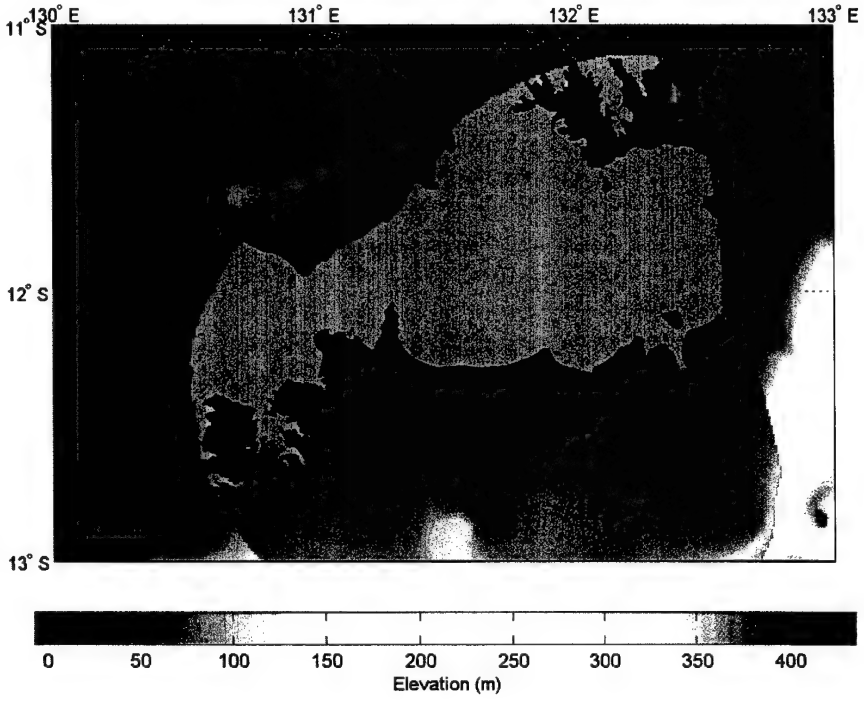


Figure 6 Line-of-sight map shown as a transparent overlay with colour set by the terrain colour map.

4.3.2 Transparent surface using arbitrary colours

Using the terrain colour map to set the colour of the line-of-sight overlay has the disadvantage of poor contrast with the terrain. But simply resetting the colour of the surface will not work, since the `NaN` values only correspond to 'no colour' if they index a colour map.

The key to producing an arbitrarily coloured overlay is to use the `alt` argument of the `meshm` function to specify where a surface should be visible (altitudes above the terrain) and invisible (altitudes below the terrain). The desired map consists of three intersecting surfaces: the terrain, the ocean, and the line-of-sight indicator which is either at altitudes above the terrain or below the ocean. The `FaceColor` property of this indicator surface is set to the desired colour and the surface then made transparent:

```
alt = vmaps;
alt = max(map(:)) .* alt;
alt(alt <= 0) = -10;
h_mesh = meshm(vmaps, vmaplegend, size(vmaps), alt);
set(h_mesh, 'FaceColor', 'red')
zdatam(handlem('allpatch'), 0)
zdatam(handlem('frame'), -1)
set(gcf,'renderer', 'OpenGL'), alpha(h_mesh, 0.5)
```

Note that it is necessary to readjust the z position of the coastal patch map and the frame (which represents the ocean), since these are plotted underneath the other surfaces by default.

By creating different `alt` matrices, more complex overlays can be created. For example, the line-of-sight map `vmaps` as can be defined as:

```
vmaps = vmapA + vmapB;
```

This allows us to determine where the limits for the two sites overlap (`vmaps == 2`). The overlapping area can then be shown as another surface, using a different colour (Figure 7).

The way in which the coloured surfaces are plotted is made more clear by a three-dimensional view (Figure 8).

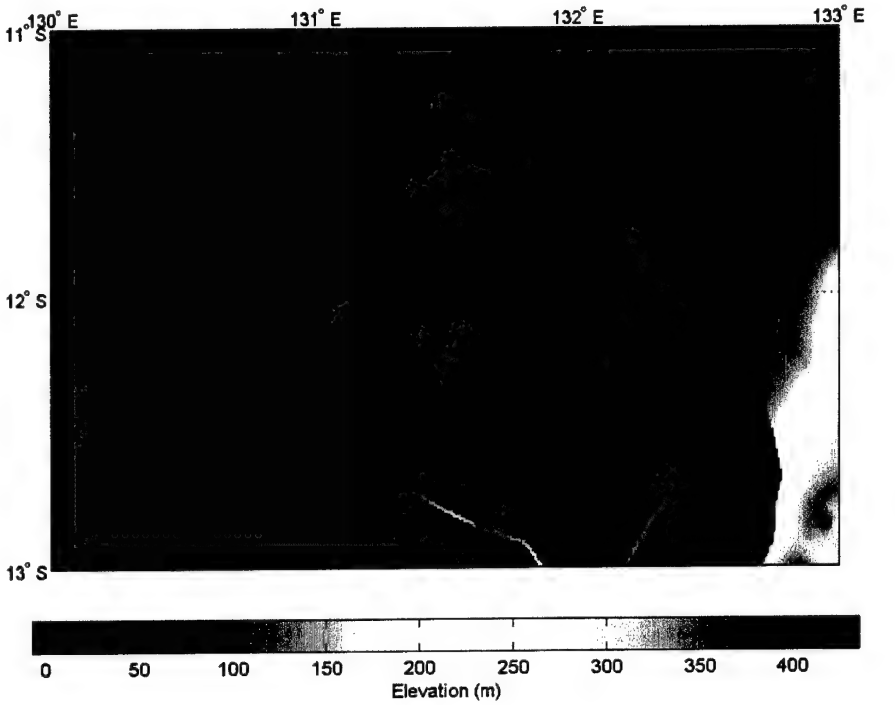


Figure 7 Line-of-sight map shown as two transparent overlays with arbitrary colours. Magenta indicates coverage by single site, red indicates overlapping coverage.

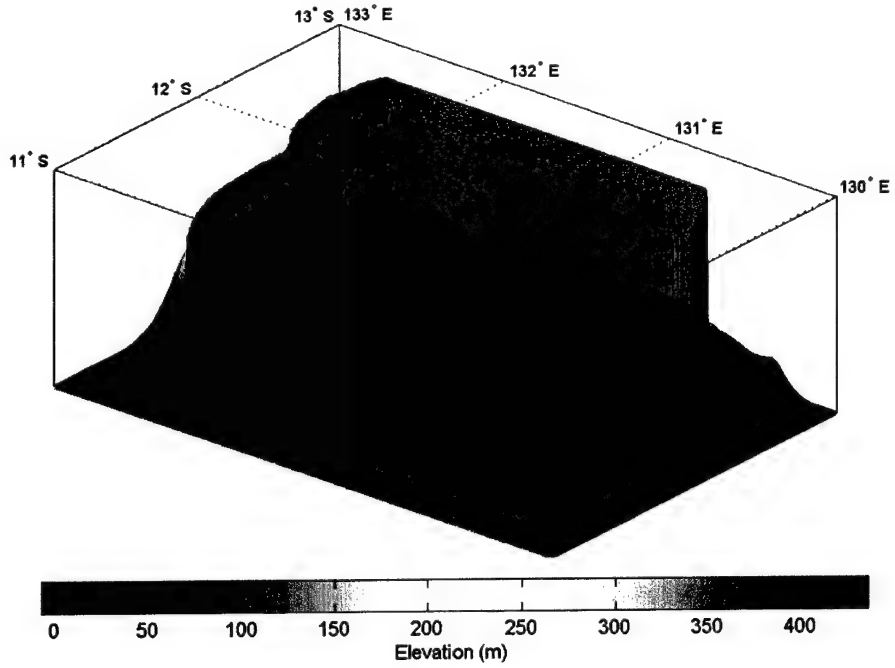


Figure 8 Three-dimensional view of line-of-sight map similar to Figure 7 (vertical scale exaggerated 200×).

4.3.3 Arbitrary colour projected onto terrain map

If the `alt` matrix contains the actual terrain data within the line-of-site limits, then the line-of-sight map can be draped over the terrain map within the limits and hidden under the terrain outside the limits:

```
alt = map;                % same as terrain map
alt(alts < 0) = 0;        % make sea level 0, not -1
alt(vmaps <= 0) = -10;    % 'masked' terrain map
h_mesh = meshm(map,maplegend,size(map),alt);
set(h_mesh, 'FaceColor', 'r')
zdatam(handlem('allpatch'),0)
zdatam(handlem('frame'), -1)
h_light = lightangle(260,50);
material dull
```

As with Figure 7, multiple surfaces can be overlaid to produce a more complex visualisation (Figure 9). Lighting effects are needed to show the terrain within the line-of-sight limits because transparency does not allow terrain features to show through properly in this case.

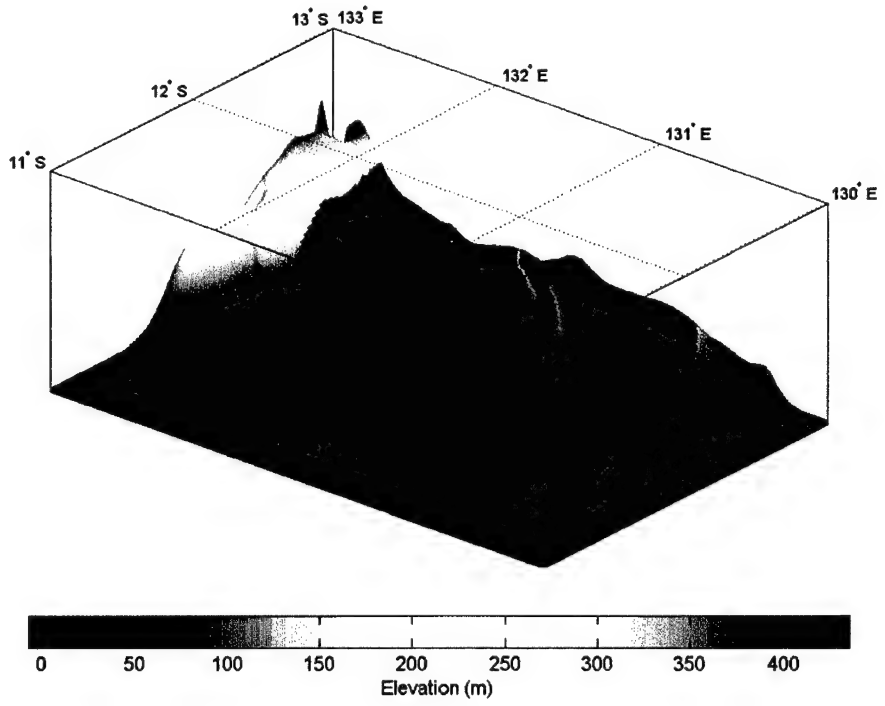


Figure 9 Line-of-sight map draped onto a three-dimensional view (vertical scale exaggerated 200×). Elevation data is shown on terrain outside the line-of-sight limits. Terrain features within the limits are highlighted by lighting effects.

4.3.4 Outline of line-of-sight

There are situations where it might be desirable to display several line-of-sight results on the same map. For example, the map might be required to illustrate the estimated performance of a radar against targets at several altitudes. One way of displaying this is to use the technique illustrated in Figure 7. Another is to plot only the line-of-sight limits (Figure 10).

Since the line-of-sight map consists of only ones and zeros, a contour map with a single contour can be used to display the line-of-sight limit on an existing map:

```
[c_matrix, h_c] = contourm(vmap, vmaplegend, [1,1]);
```

Properties such as colour and line style can then be set via the handle `h_c`.

Unfortunately, the contour may be returned as several separate line segments, i.e. several series of latitude-longitude pairs separated by `NaN` values in the contour matrix `c_matrix`. This was the case for the line marked 'Target 1' in Figure 10. Since it was not possible to plot this using a solid line, filled markers were used instead.

Artefacts such as tiny circles may appear within the line-of-sight limits, due to errors in the line-of-sight calculations. While it may be possible to identify these artefacts in the contour matrix, the cause of the artefacts can be addressed in the line-of-sight map itself. An algorithm for this is given in the Appendix.

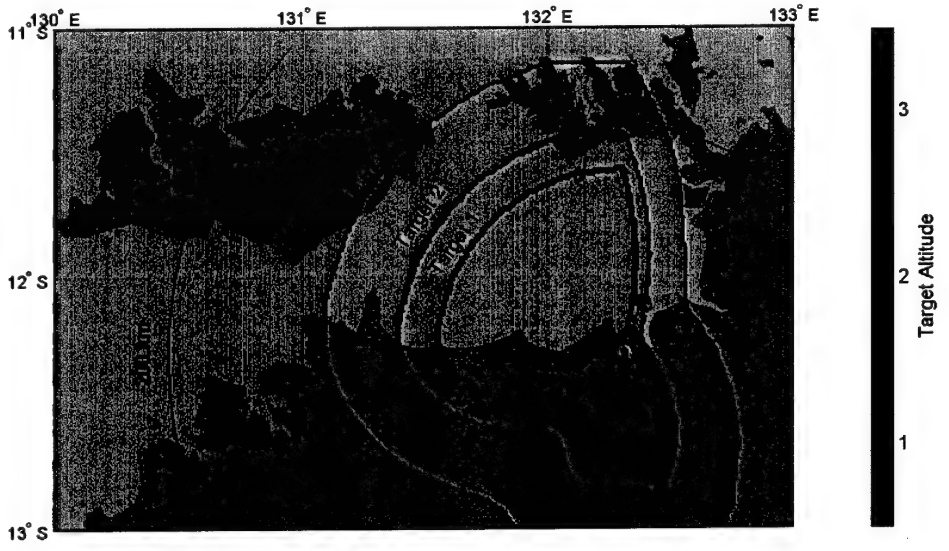


Figure 10 Line-of-sight limits for three targets at different altitudes.

The outlines can be labelled by placing text on the figure, or by using a legend or a colour bar. The colour bar is based on a colour map with each colour corresponding to a line-of-sight outline. For example:

1. The colours of the lines in Figure 10 were set from a colour map, e.g.:

```
cmap = [0.4,0,0; 0.7,0,0; 1,0,0];
set(h_targ1, 'Color', cmap(1,:))
```

2. The colour map was applied to the figure, then the colours of the patches were reset to the desired values:

```
colormap(cmap)
h_patch = handlem('allpatch');
for n = 1 : length(h_patch),
   set(h_patch(n), 'FaceColor', [0,1,0.5])
end
setm(gca, 'FFaceColor', [0.5,1,1])
```

3. The colour bar was created with a vertical label:

```
h_cb = lcolorbar({'1', '2', '3'});
set(get(h_cb,'Ylabel'), 'String', 'Target Altitude')
```

Colour bars for line-of-sight limits can only be used for displays where the terrain elevation is not colour-coded. If the limits were plotted over a terrain map such as Figure 3, the colour map would be applied to the terrain. A contour map or draped surface with lighting could be used to show terrain instead.

To use a legend, the handles of the outlines must be specified, e.g. to create a legend instead of a colour bar to the right of the map in Figure 10:

```
h_leg = legend([h_targ1,h_targ2,h_targ3],'1','2','3',-1);
set(get(h_leg,'Title'),'String','Target')
```

Without these handles, the legend might be created for other objects in the figure instead of the outlines.

4.3.5 Floating transparent surfaces using arbitrary colours

The line-of-sight limits returned by `contourm` (in `c_matrix`) can be considered to be the vertices of a polygon and plotted as such using `fill3m`:

```
clmo(h_c)   % remove line plotted by contourm
vlats = c_matrix(2,2:end);
vlons = c_matrix(1,2:end);
alt = 500;
h_patch = fill3m(vlats, vlons, alt, 'r');
set(gcf, 'renderer', 'opengl');
set(h_patch, 'LineStyle', 'none', 'FaceAlpha', 0.3)
```

This allows a similar control over appearance to the method described in §4.3.2. The difference is that in a three-dimensional view, the polygons appear to float at the specified altitude over the terrain (Figure 11) instead of columns (Figure 8). A three-

dimensional view of a surface indexing the terrain colour map (Figure 6) would look similar to Figure 11, except for the colour of the transparent surface.

If the points in the line-of-sight limits are not returned in the correct order, they can be sorted based on the azimuth from the observer location. This will not necessarily sort them into the 'correct' order if the observer location is close to the line-of-sight limit, e.g. the 'Target 1' line in Figure 10, but it may be a sufficient approximation.

1. Remove the `NaN` separators from the outline:

```
vlats = c_matrix(2,2:end); vlats(isnan(vlats)) = [];
vlons = c_matrix(1,2:end); vlons(isnan(vlons)) = [];
```

2. Calculate the bearing from the observer to each point on the outline:

```
siteLatLon = repmat([SiteLat,SiteLon], length(vlats), 1);
az = azimuth(siteLatLon, [vlats(:),vlons(:)]);
```

3. Re-order outline points based on sorted bearing:

```
[az, azOrd] = sort(az);
vlats = vlats(azOrd); vlons = vlons(azOrd);
```

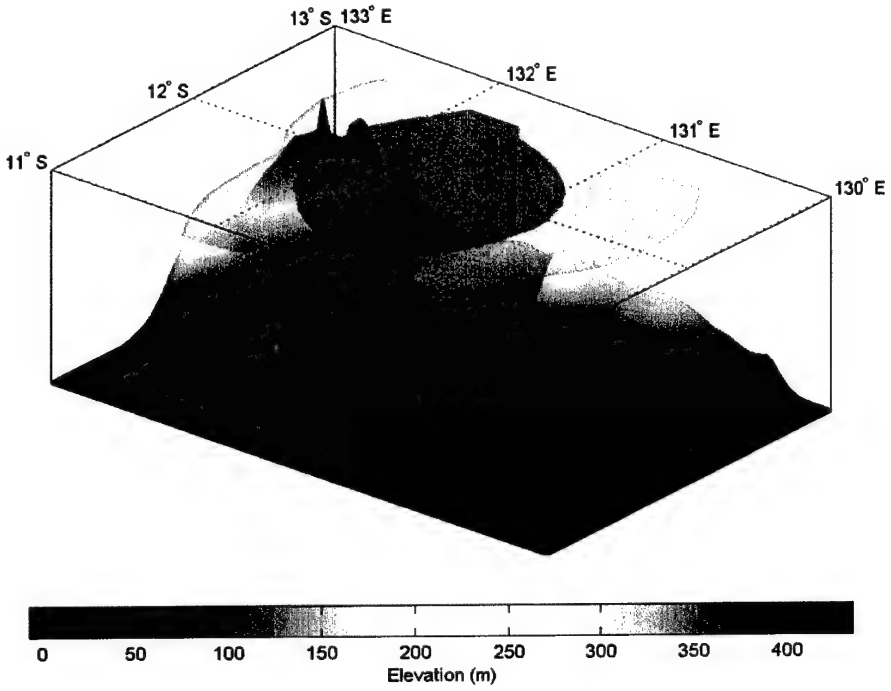


Figure 11 Line-of-sight map shown as two transparent overlays with arbitrary colours.

4.4 Distance Markers

Distance is easily displayed on maps using the `scaleruler` command. But in some situations, radial distance from a site (e.g. Figure 10) is more appropriate. This can be shown by plotting circles at specified radii from the site using `scircle1`. The number of circles that will fit on the map can be estimated based on the map legend, which specifies the location of the northwest corner of the map:

```
Deg_to_corner = distance([SiteLat,SiteLon], maplegend(2:3));
Dist_to_corner = deg2km(Deg_to_corner);
DistMarks = 100 : 100 : Dist_to_corner;  % 100 km steps
DegMarks = km2deg(DistMarks(:));
[latc, lonc] = scircle1(SiteLat, SiteLon, DegMarks);
plotm(latc, lonc, 'k')
```

The `plotm` command automatically trims off the parts of any lines that extend beyond the edges of the map (Figure 10).

4.5 Three-dimensional Displays

4.5.1 Adjustments for three-dimensional views

The `view` command or the 'Rotate 3D' button in the figure window toolbar can be used to display maps in three dimensions. But before either of these are used, it is necessary to readjust the data aspect ratio (vertical exaggeration) and vertical limits so that the map is displayed:

```
daspectm('m', 200)     % 200× vertical exaggeration
set(gca, 'ZLim', [-1, 1.2*max(map(:))])
zdatam(handlem('allpatch'), 0)
zdatam(handlem('frame'), -1)
gridm reset, mlabel('reset'), plabel('reset')
```

It may also be necessary to move the parallel and meridian labels (i.e. the latitude and longitude labels) depending on the view. The default label positions for the maps in this report were along the northern and western edges. The three-dimensional views were taken from the northwest, so the labels were moved to the southern and eastern edges. For two-dimensional views, these positions are most easily changed using `mlabel('south')` and `plabel('east')`. This does not work for a three-dimensional view though, and so the `position` properties of the labels were adjusted using the `set` command instead. Arrays of handles to the labels were obtained using `handlem('mlabel')` and `handlem('plabel')`.

4.5.2 OpenGL Issues

All displays with transparent overlays required OpenGL graphics rendering. This can cause difficulties on some computers, including:

- slow update of changes to figure;
- incorrect display of colour settings;
- difficulty in copying the figure.

The slow update of changes means that it can be frustrating to set the view using the 'Rotate 3D' button. The `view` command can be used instead, and has the advantage of allowing reproducible views (e.g. `view(220,25)` was used for figures in this document).

Colour settings may change, particularly when lighting effects are applied. A specific problem found producing the figures in this report was that the sea colour darkened considerably. It is easiest to just accept these changes, although it was found that the problem was reduced by setting the `'FaceLighting'` property of the map frame to `'Gouraud'`.

It may be more reliable to export figures (under the Figure window's File menu) than to copy them (under the Edit menu). The Portable Network Graphic format was used for all figures in this document. Even using this option, some computers would sometimes fail to export all features of the map, e.g. the markers for site positions.

5. Conclusion

Line-of sight information can be displayed in many ways:

- 'visible' and 'not visible' areas shaded on a contour map or a relief map;
- transparent surfaces above the map, which can be coloured according to the terrain map colours or some other arbitrary colour;
- surfaces draped over the terrain map;
- outlines of the line-of-sight.

The most appropriate display will depend on the application and what additional information is to be shown in the map.

Appendix A: Removing Artefacts

A.1. Artefacts in Line-of-Sight Maps

The artefacts described here are due to a problem in the line-of-sight algorithm. Other artefacts may be introduced in the line-of-sight limits by errors in terrain maps, or terrain maps with a resolution that is too coarse for the application.

The line-of-sight algorithm works by calculating the observer's viewing angles to all points along a section through the terrain, taken from the observer's position to the edge of the map. The points are marked as visible or hidden, and this information is transferred to a blank map with the same resolution as the original. This is repeated for each point along the edge of the map.

During this process, some map positions that should be marked as visible may not be set at all, due to errors in interpolating back and forth between different grid resolutions. If the line-of-sight map is displayed as a black and white image, these positions show up as 'speckles'. They become more obvious when the line-of-sight map is converted to an outline, since each speckle will be surrounded by an outline. Worse, speckles sometimes appear along short diagonal lines.

A.2. Removing Artefacts

The exact test for 'speckle' artefacts depends on whether we consider diagonal lines in the line-of-sight map to be artefacts or whether we only consider isolated points to be artefacts. The test is therefore separated into stages:

1. 'Possible speckles' (both isolated points and points in diagonal lines) are identified by a matrix cell (line-of-sight map) value different to the adjoining row and column cells, i.e.

$$\begin{matrix} x & 1 & x \\ 1 & 0 & 1 \\ x & 1 & x \end{matrix} \quad \text{or} \quad \begin{matrix} x & 0 & x \\ 0 & 1 & 0 \\ x & 0 & x \end{matrix}$$

   where it does not matter what the x values are. This test can be performed by testing each cell for inequality with its horizontal and vertical neighbours.
2. If all 'possible speckles' are to be removed, the values are changed by a logical NOT operation.
3. If only isolated points are to be removed, the values of the diagonal neighbours are compared to value of the 'possible speckle' are checked. If any diagonal neighbour has the same value as the 'speckle', then the values are not changed.

A test must be performed to check whether a cell is in the middle of the matrix or on an edge or corner, so that only existing neighbours are tested (i.e. so that indices remain within range).

DISTRIBUTION LIST

Visualising Line-of-Sight Information in Matlab

David Mewett, Damian Hall, Adam Davies

AUSTRALIA

DEFENCE ORGANISATION

	No. of copies
Task Sponsor	
Director General Intelligence Surveillance Reconnaissance and Electronic Warfare Development	1
S&T Program	
Chief Defence Scientist FAS Science Policy AS Science Corporate Management Director General Science Policy Development	shared copy
Counsellor Defence Science, London	Doc Data Sheet
Counsellor Defence Science, Washington	Doc Data Sheet
Scientific Adviser to MRDC Thailand	Doc Data Sheet
Scientific Adviser Joint	1
Navy Scientific Adviser	Doc Data Sheet
Scientific Adviser - Army	Doc Data Sheet
Air Force Scientific Adviser	Doc Data Sheet
Scientific Adviser to the DMO M&A	Doc Data Sheet
Scientific Adviser to the DMO ELL	Doc Data Sheet
Director of Trials	1
Information Sciences Laboratory	
Chief of Intelligence, Surveillance and Reconnaissance Div.	Doc Data Sheet
Research Leader Wide Area Surveillance Branch	Doc Data Sheet
Mr D. Fogg (Head SSMA Group)	1
Author(s):	
Dr D. Mewett	1
Dr D. Hall	1
Mr A. Davies	1
DSTO Library and Archives	
Library Edinburgh	1 + Doc Data Sheet
Australian Archives	1
Capability Systems Division	
Director General Maritime Development	Doc Data Sheet
Director General Information Capability Development	Doc Data Sheet
Office of the Chief Information Officer	
Deputy CIO	Doc Data Sheet

Director General Information Policy and Plans — Doc Data Sheet
AS Information Structures and Futures — Doc Data Sheet
AS Information Architecture and Management — Doc Data Sheet
Director General Australian Defence Simulation Office — Doc Data Sheet

Strategy Group
Director General Military Strategy — Doc Data Sheet
Director General Preparedness — Doc Data Sheet

HQAST
SO (Science) (ASJIC) — Doc Data Sheet

Navy
Director General Navy Capability, Performance and Plans, Navy Headquarters
Doc Data Sheet
Director General Navy Strategic Policy and Futures, Navy Headquarters
Doc Data Sheet

Army
ABCA National Standardisation Officer, Land Warfare Development Sector, Puckapunyal — e-mailed
Doc Data Sheet

SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), Enoggera QLD
Doc Data Sheet
SO (Science) - Land Headquarters (LHQ), Victoria Barracks NSW
Doc Data & Exec Summ

Intelligence Program
DGSTA Defence Intelligence Organisation — 1
Manager, Information Centre, Defence Intelligence Organisation — 1 (PDF version)
Assistant Secretary Corporate, Defence Imagery and Geospatial Organisation
Doc Data Sheet

Defence Materiel Organisation
Head Airborne Surveillance and Control — Doc Data Sheet
Head Aerospace Systems Division — Doc Data Sheet
Head Electronic Systems Division — Doc Data Sheet
Head Maritime Systems Division — Doc Data Sheet
Head Land Systems Division — Doc Data Sheet
Head Industry Division — Doc Data Sheet
Chief Joint Logistics Command — Doc Data Sheet
Management Information Systems Division — Doc Data Sheet
Head Materiel Finance — Doc Data Sheet

Defence Libraries
Library Manager, DLS-Canberra — Doc Data Sheet
Library Manager, DLS - Sydney West — Doc Data Sheet

OTHER ORGANISATIONS

National Library of Australia	1
NASA (Canberra)	1

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy	
Library	1
Head of Aerospace and Mechanical Engineering	1
Hargrave Library, Monash University	Doc Data Sheet
Librarian, Flinders University	1

OUTSIDE AUSTRALIA

INTERNATIONAL DEFENCE INFORMATION CENTRES

US Defense Technical Information Center	2
UK Defence Research Information Centre	2
Canada Defence Scientific Information Service	e-mail link to pdf
NZ Defence Information Centre	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1

SPARES	5
Total number of copies:	**30**

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Visualising Line-of-Sight Information in Matlab

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

David Mewett, Damian Hall and Adam Davies

5. CORPORATE AUTHOR

Information Sciences Laboratory
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-GD-0397	AR-013-059	General Document	March 2004

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
E9505/25/187	JTW 01/367	DGC4ISREW	21	1

13. URL on the World Wide Web

http://www.dsto.defence.gov.au/corporate/reports/DSTO-GD-0397.pdf

14. RELEASE AUTHORITY

Chief,
Intelligence, Surveillance and Reconnaissance Div.

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DEFTEST DESCRIPTORS

Matlab, Line of sight, Mapping, Computer programs

19. ABSTRACT

There are many ways to visualise areas that are within line-of-sight of a sensor or communications transmitter. This document describes how to use visualisation and mapping commands in Matlab to produce a variety of line-of-sight maps to suit customers" requirements.